Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по дисциплине: "Разработка кроссплатформенных приложений"

на тему: " Многопоточность в Java"

Выполнили:

Студенты группы 21ВВB1

Кирьянов В.Е.

Аляев А.О.

Приняли

К.т.н. Юрова О.В.

К.т.н. Карамышева Н.С.

Пенза 2024

**Название работы:** «Многопоточность в Java».

**Цель работы:** научиться создавать многопоточные приложения c использованием стандартных средств языка Java.

**Задание на лабораторную работу**

Модифицировать приложение из предыдущей лабораторной работы, реализовав вычисление определенного интеграла в нескольких дополнительных потоках (число потоков определяется номером варианта), снимая нагрузку с основного потока и предотвращая "подвисание" графического интерфейса. Варианты с номерами до 5 включительно реализуют многопоточность путем наследования от класса Thread, остальные реализуют интерфейс Runnable. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

Ход работы:

Реализовали вычисление определенного интеграла в шести дополнительных потоках.

Листинг программы.

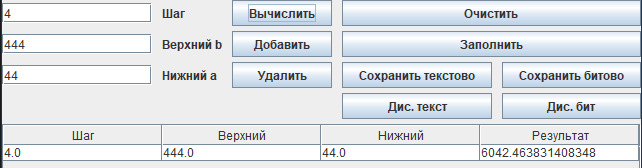
Class Main.

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
  
import static java.lang.Math.\*;  
class Res {  
 public double result = 0.0;  
 public synchronized void sum(double input\_res){ result += input\_res; }  
 public double get(){ return result; }  
}  
public class Main {  
  
 public static double Integral(double a, double b, double step) throws InterruptedException {  
 Res result = new Res();  
  
 double h = b - a; //интервал  
 h = h / 6; //interval segmenta  
 b = a + h;  
 //double stepCount = 0;  
 //if((h/step) % 1 == 0){ stepCount = h / step; }  
 //else{ stepCount = floor(h / step); } //2 //2.5  
 List<Thread> normalList = new ArrayList<Thread>();  
 List<Thread> threads = Collections.*synchronizedList*(normalList);  
 for(int i = 1; i<7; i++){  
 threads.add(new Thread(new MyThread(a, b, step, h, result), String.*valueOf*(i)));  
 a += h;  
 b += h;  
 }  
 for(Thread temp: threads){  
 temp.start();  
 temp.join();  
 }  
  
// for (int i : mass) {  
// myThread = new Thread(new MyThread(a, b, step, h, result), String.valueOf(i));  
// myThread.start();  
// }  
// double left = a;  
// double right = a + step;  
// double res = 0.0;  
  
// for(double i = 0.0; i <= stepCount ; i++) {  
  
// double a1 = sqrt(left);  
// double b1 = sqrt(right);  
// res = res + (double) 1 / 2 \* step \* (a1 + b1);  
  
// left = left + step;  
// if(b < (right + step)){break;}  
// right = right + step;  
// }  
// if(b < (right + step)){  
// right = b;  
// step = b - left;  
// double a1 = sqrt(left);  
// double b1 = sqrt(right);  
// res = res + (double) 1 / 2 \* step \* (a1 + b1);  
// }  
 //return res;  
 return result.get();  
 }  
}

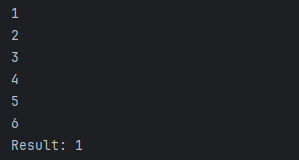
Class MyThread:

import static java.lang.Math.floor;  
import static java.lang.Math.sqrt;  
  
public class MyThread implements Runnable{  
 double a;  
 double b;  
 double step;  
 double h;  
 double stepCount = 0;  
 double left;// = a;  
 double right;// = a + step;  
 double res = 0.0;  
 Res result;  
 public MyThread(double a, double b, double step, double h, Res result){  
 this.a = a;  
 this.b = b;  
 this.step = step;  
 this.h = h;  
 this.result = result;  
 this.left = a;  
 this.right = a + step;  
 }  
  
 @Override  
 public void run() {  
  
 if((h/step) % 1 == 0){ stepCount = h / step; }  
 else{ stepCount = floor(h / step); } //2 //2.5  
  
 for(double i = 0.0; i <= stepCount ; i++) {  
  
 double a1 = sqrt(left);  
 double b1 = sqrt(right);  
 res = res + (double) 1 / 2 \* step \* (a1 + b1);  
  
 left = left + step;  
 if(b < (right + step)){break;}  
 right = right + step;  
 }  
 if(b < (right + step)){  
 right = b;  
 step = b - left;  
 double a1 = sqrt(left);  
 double b1 = sqrt(right);  
 res = res + (double) 1 / 2 \* step \* (a1 + b1);  
 }  
 System.out.printf("%s \n", Thread.currentThread().getName());  
 result.sum(res);  
  
 }  
}

Результаты программы.



Выводит номера потоков и затраченное время.



Вывод: научились создавать многопоточные приложения c использованием стандартных средств языка Java.